线性表的概念

线性表是最基本、最简单、也是最常用的一种数据结构。线性表（linear list）是数据结构的一种，一个线性表是n个具有相同特性的数据元素的有限序列。

特点：

1．集合中必存在唯一的一个“第一元素”。

2．集合中必存在唯一的一个 “最后元素” 。

3．除最后一个元素之外，均有唯一的后继(后件)。

4．除第一个元素之外，均有唯一的前驱(前件)。

线性表包括顺序表和链表，顺序表里面元素的地址是连续的。链表里面节点的地址不是连续的，是通过指针连起来的。

顺序表的概念

顺序表是在计算机内存中以数组的形式保存的线性表，线性表的顺序存储是指用一组地址连续的存储单元依次存储线性表中的各个元素、使得线性表中在逻辑结构上相邻的数据元素存储在相邻的物理存储单元中，即通过数据元素物理存储的相邻关系来反映数据元素之间逻辑上的相邻关系，采用顺序存储结构的线性表通常称为顺序表。顺序表是将表中的结点依次存放在计算机内存中一组地址连续的存储单元中。

1. 关于时间，数组的调用元素是 O(1)，查找元素是O(n)，插入元素是O(n)
2. 关于空间，O(n)

由于数组是固定大小的，所以我们可以采用vector动态顺序表，时间空间同上。
所以顺序表对于插入与删除频繁的存储，时间复杂度太大

链表的概念

链表是一种物理存储单元上非连续、非顺序的存储结构，数据元素的逻辑顺序是通过链表中的指针链接次序实现的。链表由一系列结点（链表中每一个元素称为结点）组成，结点可以在运行时动态生成。每个结点包括两个部分：一个是存储数据元素的数据域，另一个是存储下一个结点地址的指针域。 相比于线性表顺序结构，操作复杂。由于不必须按顺序存储，链表在插入的时候可以达到O(1)的复杂度，比另一种线性表顺序表快得多，但是查找一个节点或者访问特定编号的节点则需要O(n)的时间，而线性表和顺序表相应的时间复杂度分别是O(logn)和O(1)。
由于不是连续的空间，所以我们需要记录表头，通过表头来往后遍历
链表对于频繁的查询，时间复杂度太大

链表通过每个结点的链域将线性表的n个结点按其逻辑顺序链接在一起的，每个结点只有一个链域的链表称为单链表（Single Linked List）。
也就是说我们只知道当前元素下一个元素是谁，但是不知道上一个元素是谁，所以我们遍历的时候只能通过表头从头往后遍历。

双向链表也叫双链表，是链表的一种，它的每个数据结点中都有两个指针，分别指向直接后继和直接前驱。所以，从双向链表中的任意一个结点开始，都可以很方便地访问它的前驱结点和后继结点

如何模拟链表

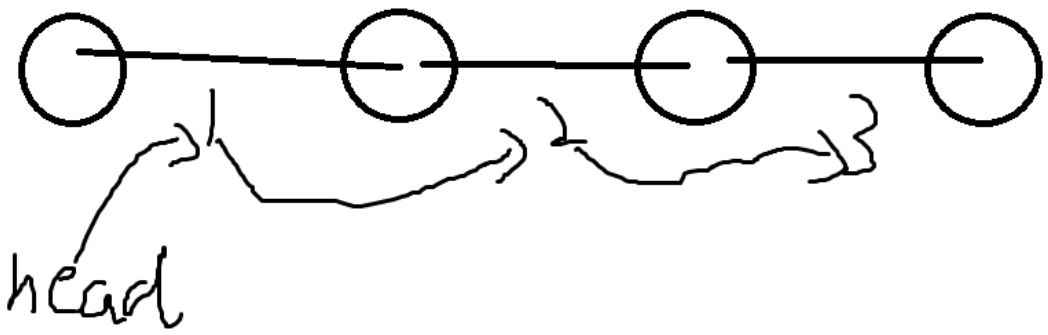
对于链表前后两个元素，我们可以抽象成两个点由一条边相连接。

```
to[x] = y; //点x后边是点y
last[y] = x; //点y前边是点x

void add(int x,int y){ //把点y添加到点x后边
    last[to[x]] = y;
    to[y] = to[x];
    to[x] = y;
}

void del(int x){ //删除点x后边的那个数
    last[to[to[x]]] = x;
    to[x] = to[to[x]];
}
```

但是我们发现这样子去写链表的前提是没有重复的元素，如果出现了多个元素，记录就已经混乱了。
因为点之间是一对多的，所以不能同时记录多个点
我们思考能否给边一个编号，用此编号来记录



用表头指向第一条边，第一条边指向第二条边，第二条边指向第三条边，我们发现这样子也是可行的，并且可以记录当前边所连接的两个点。

```
//向表头插入一个数x
void add(int x){
    ++cnt; //代表边的编号
    w[cnt] = x;
    next[cnt] = head; //下一条边是谁
    head = cnt; //表头的边编号
}

//遍历链表
for(int i = head;; i = last[i])

//删除第 k 个插入的数后面的数;
void del(int k){
    next[k] = next[next[k]];
}

//在第 k 个插入的数后面的数;
void add2(int k,int x){
    ++cnt;
    w[cnt] = x;
    next[cnt] = next[k];
    next[k] = cnt;
}
```